REMARKS

Claims 1-33 are currently pending in the application. Claims 1, and 32-33 have been amended. Claims 2-31 remain unchanged. Applicants thank the Examiner for the indication of allowability with respect to claims 21-23.

I. 103 Rejection

1. Claims 1-6, 13-17, 20, 24, 32-33 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,241,648 issued August, 31 1993 to Cheng et al. (hereafter Cheng) in view of U.S. Patent 4,905,138 issued February 27, 1990 to Borne (hereafter Borne).

A. The hybrid join technique of the Cheng reference joins two tables of relational data base management systems in a three-stage process. Initially, it is asserted that to practiced this solution requires an index to exist on one or more of the join columns of the inner table when the join predicates are combined by ANDing or that indexes on each join column of the inner table exist where index ANDing or ORing can be performed to achieve the result of an index on all join columns.

Stated formally, these pre-conditions are as follows:

1. If the join predicates are in the form of (P.sub.1 AND P.sub.2 AND . . . P.sub.n) then the solution requires an index exists on one or more join columns of the inner table where P.sub.i, i=1, . . . , n is the join predicate involving one column of each table;

2. If the join predicates are in the form of (P.sub.1 OR P.sub.2 OR . . . P.sub.n) then the solution requires that enough indexes be provided so that all the join columns of the inner table are represented (individually or collectively) in these indexes; and

3. If the join predicates are compound predicates which combine conjunctive (AND) and disjunctive (OR) forms as provided in the first and second conditions, then the first and second

conditions must hold for the corresponding forms.

In the first stage of the process of the Cheng reference, the index on the inner table's join column is scanned for rows of the inner table having join column values matching such values of rows in the outer table. This is done in response to a single pass through the outer table, which is ordered (by sorting, scanning an index on the join column of the outer table, or scanning a table via a clustered index) on the join column, and through the inner table's index on the join column (which is inherently ordered). At the end of this stage, a temporary work table containing the selected inner table RID's concatenated to their matching outer table rows is produced.

In the second stage, the temporary work table is ordered by RID, if not already in such order, or nearly so.

Last, the RID list of inner table rows is used to fetch the selected rows of the inner table. The local predicates and join predicate are applied to these rows, and those satisfying both are combined with their matching table rows and returned for placement in the join table.

A primary object of the Cheng refernce is to provide a join technique which uses existing data base structures to enhance nested loop and sort/merge join techniques. Another object of the Cheng refernce is to enable multiple processes to process data in parallel or in a pipeline, thereby reducing response time.

For example, Cheng specifically discloses:

> joins two tables of relational data base management systems in a three-stage process.

(Cheng, col. 4, lines 40-42) emphasis added).

Cheng further discloses:

TABLE I

OPEN CURSOR PROCESSING

100 Open a scan providing sorted (or mostly sorted) access to T1.
101 Open a scan on the T2 index.
. . .

```
107  Build the composite rows (T2 RID, T1 data)
     in the intermediate result table.
107a Send to sort the intermediate result table on
     T2 RID.
108  End.            . . .

123  "AND" the RIDs with those in the intermediate
     result table.
125  If EOF (on either T1 or T2), Then
126  Close the scan on T1.
127  Close the scan on the T2 index.
130  Open a sequential prefetch scan on T2.
131  Open a sequential prefetch scan on the intermediate
     result table.
```

TABLE II

FETCH CURSOR PROCESSING

```
200  Do until (row-returned .vertline. EOF).
201  Fetch the next row in the intermediate result table.
202  If the T2 RID is the same as the last one, Then
                     . . .
211  End.
```

(Cheng, column 6, line 55 to column 7, line 30) emphasis added.

In contrast , claim 1 of the instant application recites "[a] method for pipelining a table function in a database system, comprising: a) performing a set up operation when the table function is called, the table function being a user-defined function that can produce rows of data and can be used in selection, iteration, or aggregation database query language statements; b) fetching a subset of output data from a data producer; c) sending the subset of the output data to a first consumer of the output data, wherein the first consumer is the table function; d) repeating steps b) and c) until all the output data has been fetched from the data producer." Applicants submit that the hybrid join technique of Cheng is not the same as the table function of the instant application.

For example, the hybrid join technique of Cheng employs standard database structures, such as tables which require indexes, being operated on by standard system operators, such as AND, OR, and join. In contrast, the instant application is a method for pipelining a table function which is "a user-defined function that can produce rows of data and can be used in selection, iteration, or aggregation database query language statements." The tables, join operations, ANDs, and ORs of Cheng are standard system objects and operators, not user-defined table functions. As the hybrid join technique of Cheng is not, and does not employ any, table function(s) that are user-defined, produce rows as output, and can be used in selection, iteration or aggregation statements, Cheng does not disclose "[a] method for pipelining a table function in a database system, comprising: a) performing a set up operation when the table function is called, the table function being a user-defined function that can produce rows of data and can be used in selection, iteration, or aggregation database query language statements; b) fetching a subset of output data from a data producer; c) sending the subset of the output data to a first consumer of the output data, wherein the first consumer is the table function; d) repeating steps b) and c) until all the output data has been fetched from the data producer" as recited in amended claim 1.

Bourne does not cure this deficiency. Objectives of the Bourne solution include: to provide an interpreter with artificial intelligence capabilities; to provide a meta-interpreter capable of receiving a message, analyzing the message for meaning using an application dependent grammar and rules, and perform an action based on the meaning; to provide an interpreter with program and data generation capability where the programs and data are represented in a tabular data structure suitable for manipulation using ordinary database processing techniques; and to provide a meta-interpreter which is capable of building and using computer languages within the meta-interpreter.

The above-mentioned objectives can be attained by the implementation of one or more interpreters as programs dedicated to a particular task or by using a command language executed by a general purpose program which executes the functions of an interpreter as specified in a command language program. Each interpreter operates within a workspace that includes both data and programs represented as tables. Each workspace or interpreter can communicate to other workspaces, other non-interpreter processes and exterior processes or machines in a message format suitable for the message receiver. The meaning of each received message can be determined and used to govern how the interpreter responds to the message. Each interpreter uses a general table driven parser that examines a message using grammar and lexical tables to produce a parse table. The parse table is compared to data needed in a semantics table to fire one or more conditional rules resulting in meaning being applied to the message. The firing of a rule causes a function table to be evaluated. The function table includes system function calls which can perform functions chosen by a user thereby performing some action based on the meaning of the message. Among the system functions is included a Generate function which will take the contents of a table and turn it into a message and route the message to a destination where the destination can be a table, process, or device. Another system function is an Update function which will use the meaning of the message to update a database. Other system functions include Evaluate, and Expand.

For example, Bourne specifically discloses:

> Most system functions operate on all of the different pathname forms. To simplify the execution of the function, each pathname form is assigned a unique number.
>
> For example, the Delete function...takes a single argument - a pathname as in, delete, table or delete,table;entry or delete,table:entry::field, or delete,table::field.

(Bourne, column 9, lines 26-32) underline added.

Bourne further discloses:

> Referring to FIG. 3, once parsing is complete, semantic attachment with respect to a message is started by an Evaluate function calling an Expand function. At the end of the semantic attachment, the Expand function recursively calls the Evaluate function to cause the action or translation to occur. The Evaluate function that begins the semantic attachment phase is essentially a subroutine call which accesses the named table and performs the designated function. It is possible, in an alternative embodiment, to eliminate the first call to the Evaluate function and go directly to the Expand function after the message has been parsed. . . . .
>
> The Expand function implements the non-procedural rule based processing in the interpreter that provides the intelligence capability. One table (in this example the parse table, however, it could be any other table or tables designated by the user) supplies the data to be applied to a set of rules which can cause one or more rules to fire. When a rule fires, the function designated in the rule is executed. That is, the Expand function expects the name of a data table and the name of a semantics or rule table and will fire the rules with conditions that match the data.
>
> Most prior art rule based systems are a collection of if statements, however, the rules in the present invention are in the form of calls to the Evaluate function to be discussed in detail later. The function table named in the rule is executed (evaluated) when there are entries in the data table that match each argument in the function call or rule. This approach is a known as pattern directed function

call. Since each rule is tested, if no matches within the rule table exist, an optional "otherwise" function can be used as a mechanism for handling errors.

When the Expand function is executed using the Parse table 150 of FIG. 18, as the data table and the Rules table 152 of FIG. 19 as the semantics table, the following actions are performed. The Expand function scans the first rule R1 to determine whether to execute a call of the Evaluate function using the function or table called Furnace Update by determining whether the Parse Table includes a row labeled "verb" having a value equal to "set" and in addition, has rows called noun-1, noun-2 and decimal. If all the data items are available, that is, if all the data exists and satisfies the conditions associated with each argument, and in this case they do, the Evaluate function is executed (the rule fires) using the Furnace Update table. Prior to calling the Evaluate function, the values in the Parse Table are assigned to the appropriate types and a parameter table called a Substitution table 154, as illustrated in FIG. 20, is built. The parameter table or arguments list is used when executing the Furnace Update. When the interpreter returns from the function call, the next rule is examined to see if it should fire.

(Bourne, column 13, line 59 to column 14, line 57) underline added.

These passages and the explanation of Bourne teach a plurality of tables of system functions. Each of Bourne's tables can contain whichever system functions are desired. As mentioned above an interpreter decides which of the function tables to execute based on the content of the message. Some example of Bourne's system functions are "Delete", "Evaluate", and "Expand". These system functions of Bourne are not the table function of the instant application which is "a user-defined function that can produce rows of data and can be used in selection, iteration, or aggregation database query language statements." Bourne does not show a user-defined function, a function that produces rows of data, or a function that can be used in selection, iteration and aggregation statements. As such Bourne does not cure the deficiency of Cheng.

Cheng and Bourne, neither alone nor together, disclose, teach or suggest all the elements of amended claim 1. As such, Cheng and Bourne, neither alone nor together can be used to preclude patentability of claim 1 under 35 U.S.C. § 103.

B. Claims 32-33 recite substantially the same elements as claim 1, and are therefore, patentable over Cheng and Bourne for at least the same reasons.

C. Claims 2-6, 13-17, 20, and 24 depend on claim 1 and are patentable over Cheng and Bourne for at least the same reasons.

2. Claims 7-10 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Cheng and Bourne as applied to claim 1 and in further view of U.S. Patent 6,052,699 issued April 18, 2000 to Huelsbergen et al. (hereafter Huelsbergen). As shown above Cheng and Bourne do not disclose; teach, or suggest amended claim 1. Huelsbergen does not cure this deficiency.

The Heulsbergen reference provides a garbage collection technique for the concurrent operation of a mutator and garbage collector (e.g., marker and sweeper) without requiring fine-grain synchronization or atomicity amongst the mutator, marker and sweeper. In accordance with the Huelsbergen reference, three threads are used for concurrently executing the mutator, marker and sweeper. The inventive garbage collector operates through a series of so called epochs (i.e., individual garbage collection phases) wherein the collector operates in each epoch to concurrently (1) run the mutator; (2) mark all objects that were reachable (i.e., allocated) in the previous epoch with the present epoch's color; and (3) reclaim any objects marked as garbage.

For example, Huelsbergen specifically discloses:

> The present invention provides a garbage collection technique for the concurrent operation of a mutator and garbage collector

(Huelsbergen, column 5, lines 45-47) underline added.

Huelsbergen's garbage collection technique deals with background memory allocation in a database environment. Huelsbergen does not disclose a table function that is "a user-defined function that can produce rows of data, and can be used in selection, iteration, or aggregation database query language statements" and as such it does not cure the deficiencies of Cheng and Bourne. As such, Cheng, Bourne, and Huelsbergen, neither alone, nor together, disclose, teach, or suggest the limitations in claim 7-10 which depends on claim 1 and cannot be used to preclude patentability of those claims under 35 U.S.C. § 103.

3. Claims 11-12, 18-19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Cheng and Bourne as applied to claim 1 and further view of U.S. Patent 5,241,648 issued 25 April 1995 to Danneels et al. (hereafter Danneels). As shown above, Cheng and Bourne do not

disclose, teach, or suggest all the limitations in amended claim 1. Danneels does not cure this deficiency.

The Daneels reference is a method and system for loading a library requested by a service requester of an application program in a computer system. The system comprises a first loader module and a second loader module. The first loader module receives a request from the service requester to load the library, where the first loader module is part of the executable application program. The second loader module receives the request from the first loader module and loads the library, where the second loader module is an executable distinct from the executable application program.

The Danneels reference is also a method and system for loading libraries in a computer of a multicast system. The system comprises a global dynamic loader and a global dynamic loader executable. A media services manager of a multicast application program issues a request to load a media service provider library. The global dynamic loader receives the request, where the global dynamic loader is part of the executable multicast application program. The global dynamic loader executable receives the request from tile global dynamic loader and loads the media service provider library, where the global dynamic loader executable is an executable distinct from the executable multicast application program.

Danneels teaches a method and system for loading a library requested by a service requester of an application program in a computer system. For example, Danneels specifically discloses:

> According to a preferred embodiment, the present invention is a method and system for loading a library requested by a service requester of an application program in a computer system.

Danneels, column 2, lines 30-33) underline added.

Danneels further discloses:

> To establish a connectionless data transfer session, the server and a client each call the DLM_BeginSession and DLM_RegisterSocket functions to their respective local DLMs. The Local DLM responds by calling the DLM Session Callback function with the REGISTER COMPLETE event to notify the server/client that the socket has been successfully registered.

(Danneels, column 21, lines 37-43) underline added.

These passages and the explanation of Danneels teach a method and system for loading a requested library and calling different system functions one of which is named "DLM Session Callback" function. Danneels does not disclose a table function that is "a user-defined function that can produce rows of data, and can be used in selection, iteration, or aggregation database query language statements" and as such it does not cure the deficiencies of Cheng and Bourne. As such, Cheng, Bourne, and Danneels, neither alone, nor together, disclose, teach, or suggest the limitations in claim 1 on which claims 11-12, 18-19 depend and therefore cannot be used to preclude patentability of those claims under 35 U.S.C. § 103.

4. Claims 25-28 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Cheng and Bourne in further view of U.S. Patent 4,803,613 issued 7 February 1989 to Kametani et al. (hereafter Kametani). As shown above, Cheng and Bourne do not disclose, teach, or suggest all the limitations in amended claim 1. Kametani does not cure this deficiency.

<u>An object of the Kametani solution is to provide a general-purpose control apparatus</u> which has a high degree of flexibility and expansibility. Another object of the Kametani solution is to provide a hierarchical multiple controller system which has a high degree of flexibility and expansibility. A further object of the Kametani solution is to provide a decentralized control apparatus provided with a plurality of intelligent slave modules and a host processor for supervising the plane modules wherein the control process by the apparatus advances under the leadership of the slave modules.

<u>The control apparatus of the Kametani solution for controlling an object including a plurality of controlled elements is provided with a host processer, a plurality of slave modules and a communication network for establishing communications between the host processor and the plurality of slave modules.</u> The host processor supervises the whole apparatus and interprets a control program in response to a demand from each of the slave modules and generates a command for controlling the controlled object. Each of the slave modules is allotted to one of the controlled elements and has its own processor which interprets and executes commands for the controlled element allotted thereto, and upon completion of the execution of each command, generates a demand for a further command. Thus, through the above communication network, <u>each command is transmitted from the host processor to the related slave module and each</u>

demand is transmitted from each slave module to the host processor.

For example, Kametani specifically discloses:

> The host processor supervises the whole apparatus and interprets a control program in response to a demand from each of the slave modules and generates a command for controlling the controlled object. Each of the slave modules is allotted to one of the control elements and has its own processor which interprets and executes commands for the controlled elements allotted thereto, and upon completion of each command, generates a demand for a further command. Thus, through the above communication network, each command is transmitted from the host processor to the related slave module and each demand is transmitted from each slave to the host processor.

(Kametani, column 2, lines 21-36) underline added.

This passage and the explanation of Kametani teach a master/slave apparatus for equipment control. Kametani does not disclose a table function that is "a user-defined function that can produce rows of data, and can be used in selection, iteration, or aggregation database query language statements" and as such it does not cure the deficiencies of Cheng and Bourne. As such, Cheng, Bourne, and Kametani, neither alone, nor together, disclose, teach, or suggest the limitations in claim 1 on which claims 25-28 depend and therefore cannot be used to preclude patentability of these claims under 35 U.S.C. § 103.

5. Claims 29-31 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Cheng and Bourne in further view of U.S. Patent 5,937,415 issued 10 August 1999 to Sheffield et al. (hereafter Sheffield). As shown above, Cheng and Bourne do not disclose, teach, or suggest all the limitations in amended claim 1. Sheffield does not cure this deficiency.

The Sheffield solution describes a Client/Server Database System with improved methods for performing data transfers. The system includes one or more Clients (e.g., Terminals or PCs) connected via a Network to a Server. In general operation, Clients store data in and retrieve data from one or more database tables resident on the Server by submitting SQL commands, some of which specify "queries"--criteria for selecting particular records of a table.

The Sheffield system implements a "Data Pipeline" feature for programming replication of data from one database to another in client applications. Specifically, a pipeline object and SQL SELECT statement are built using a Pipeline Painter. The Data Pipeline lets a user (developer) easily move data from a high-end database server (e.g., Sybase) to a local database (Watcom SQL), all without the user having to issue SQL commands. The pipeline object facilitates moving data from one database management system to another, or between databases of the same type.

A method of the Sheffield solution for copying (or moving) data from a source data source to a destination data source, comprises selecting source and destination database profiles for the source and destination data sources, each profile comprising sufficient information for supporting a connection to a respective one of the data sources, establishing a connection to the source and destination data sources, displaying a graphical user interface for graphically defining a pipeline object, the pipeline object specifying copying of data from the source data source to the destination data source, saving the pipeline object to a storage device, and copying data from the source data to the destination data by executing the pipeline object.

For example, Sheffield specifically discloses:

> A Client/Server Database System with improved methods for performing data transfers is described. The system includes one or more Clients (e.g., Terminal if PCs) connected via a Network to a Server. In general operation, Client store data in an retrieve data from one or more database tables resident on the Server by submitting SQL commands, some of which specify "queries" --criteria for selecting particular records of a table.
> The system implements a "Data Pipeline" feature for programming replication of data from one database to another in client applications. Specifically, a pipeline object and SQL SELECT statement are built using a Pipeline Painter. The Data Pipeline lets a user (developer) easily move data from a high-end data base server (e.g. Sybase) to a local database (Watcom SQL), all without the user having to issue SQL commands.

(Sheffield, column 2, lines 42-57) underline added.

Sheffield further discloses:

> The Optimizer, therefore, performs an analysis of the query and picks the best execution plan, which in turn results in particular ones of the Access Methods being invoked during query execution.

(Sheffield, column 7, lines 26-32) underline added.

These passages and the explanation of Sheffield show a client/database system for replicating data from one database system to another and an optimizer that chooses the best query execution plan. Sheffield does not disclose a table function that is "a user-defined function that can produce rows of data, and can be used in selection, iteration, or aggregation database query language statements" and as such it does not cure the deficiencies of Cheng and Bourne. As such, Cheng, Bourne, and Sheffield, neither alone, nor together, disclose, teach, or suggest the limitations in claim 1 on which claims 25-28 depend and therefore cannot be used to preclude patentability of those claims under 35 U.S.C. § 103.

6. Further, even if the main references of Cheng and Bourne and the secondary references of Huelsbergen, Danneels, Kametani, and Sheffield possessed all the recited elements, there is no motivation to combine them all. Cheng is directed to a hybrid join technique for joining two tables. Bourne is a message interpreter for determining what group of system functions to execute. Huelsbergen is directed to a garbage collection technique for the concurrent operation of a mutator and a garbage collector. Danneels is directed towards a method and system for loading a library requested by a service requester of an application program in a computer system. Kametani is directed to a master/slave apparatus for equipment (e.g., robot) control. Sheffield is directed to Client/Server system with improved methods of replicating data from one database system to another. There is no motivation to combine the hybrid join technique of Cheng with the message interpreter of Bourne, with the garbage collection of Huelbergen, with the system for loading library requests of Danneels, with a master/slave apparatus for equipment control of Kametani and with the improved methods of data replication between databases of Sheffield.

CONCLUSION

On the basis of the above remarks, reconsideration and allowance of the claims is believed to be warranted and such action is respectfully requested. If the Examiner has any questions or comments, the Examiner is respectfully requested to contact the undersigned at the number listed below.

The Commissioner is authorized to charge any fees due in connection with the filing of this document to Bingham McCutchen's Deposit Account No. **50-2518,** referencing billing number **7011412001**. The Commissioner is authorized to credit any overpayment or to charge any underpayment to Bingham McCutchen's Deposit Account No. **50-2518,** referencing billing number **7011412001**.

Respectfully submitted,
Bingham McCutchen LLP

Dated: April 5, 2005

By: Janet D. Chance
Janet D. Chance
Reg. No. 55,048

Three Embarcadero Center, Suite 1800
San Francisco, CA 94111-4067
Telephone: (650) 849-4904
Telefax: (650) 849-4800